Coding

1. Loading the dataset

```
import numpy as np

from keras.datasets import cifar10
from keras.utils.np_utils import to_categorical
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
```

2. Examining the dataset

```
print("Shape of training data:")
print(X_train.shape)
print(y_train.shape)
print("Shape of test data:")
print(X_test.shape)
print(y_test.shape)
import matplotlib.pyplot as plt

cifar_classes = ['airplane', 'automobile', 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck']
print('Example training images and their labels: ' + str([x[0] for x in y_train[0:5]]))
print('Corresponding classes for the labels: ' + str([cifar_classes[x[0]] for x in y_train[0:5]]))

f, axarr = plt.subplots(1, 5)
f.set_size_inches(16, 6)

for i in range(5):
img = X_train[i]
axarr[i].imshow(img)
plt.show()
```

3. Preparing the Dataset

```
# Transform label indices to one-hot encoded vectors

y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)

# Transform images from (32,32,3) to 3072-dimensional vectors (32*32*3)

X_train = np.reshape(X_train,(50000,3072))
```

```
X_test = np.reshape(X_test,(10000,3072))
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')

# Normalization of pixel values (to [0-1] range)

X_train /= 255
X_test /= 255
```

CNN classifier

1. Preparing the dataset

```
(X_train, y_train), (X_test, y_test) = cifar10.load_data()
y_train = to_categorical(y_train, num_classes=10)
y_test = to_categorical(y_test, num_classes=10)
X_train = X_train.astype('float32')
X_test = X_test.astype('float32')
X_train /= 255
X_test /= 255

print("Shape of training data:")
print(X_train.shape)
print(y_train.shape)
print("Shape of test data:")
print(X_test.shape)
print(y_test.shape)
```

2. Creating CNN model

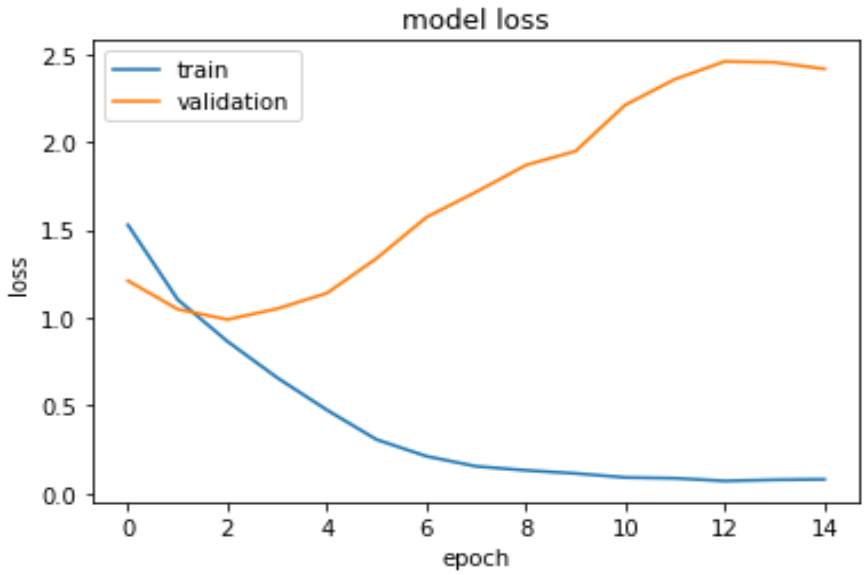
```
from keras.layers import Dense, Flatten

from keras.layers import Conv2D, MaxPooling2D

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))

model.add(Conv2D(32, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Flatten())
```

```
model.add(Dense(256, activation='relu'))

model.add(Dense(10, activation='softmax'))

sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)

model.compile(loss='categorical_crossentropy', metrics=['accuracy'], optimizer=sgd)
```

3. Training the CNN

```
history = model.fit(X_train, y_train, batch_size=32, epochs=15, verbose=2,
validation_split=0.2)
```

4. Plot Losses(history)



5. Evaluating the CNN

```
score = model.evaluate(X_test, y_test, batch_size=128, verbose=0)
print(model.metrics_names)
print(score)
```